AMENDMENTS TO THE CLAIMS

The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (currently amended) A computer-implemented method of identifying compatible software threads to execute on a Simultaneous Multi-Threading (SMT) ~~an SMT~~ processor, said method comprising:

identifying a time interval during which both a first thread and a second thread are executing on the SMT processor;

retrieving a performance value that is a cycles per instruction (CPI) value and that occurred during the identified time interval;

determining, based upon the retrieved performance value, whether the first thread is compatible with the second thread, wherein the determining includes comparing the CPI value to a threshold value, wherein the first thread and second thread are determined to be compatible if the CPI value is better than the threshold value; and

recording the compatibility of the first thread with the second thread in response to the determination.

2. (canceled)

3. (currently amended) The method as described in claim 1 [[2]] wherein the CPI value further comprises:

retrieving a number of cycles value indicating the number of processing cycles that occurred during the time interval;

retrieving a number of instructions value indicating the number of instructions that were executed by the SMT processor during the time interval; and

dividing the number of cycles value by the number of instructions value, the dividing resulting in the CPI value.

4. (canceled)

5. (currently amended) The method as described in claim 1 [[2]] further comprising:

writing a first identifier corresponding to the first thread and the CPI value to a compatibility list that corresponds to the second thread.

6. (original) The method as described in claim 5 wherein the writing is performed in response to identifying an empty field in the second thread's compatibility list.

7. (original) The method as described in claim 5 wherein the writing is performed in response to: behavior

comparing the CPI value to one or more previously recorded CPI values that correspond to one or more previously identified compatible threads; and

determining that the CPI value is better than at least one of the previously recorded CPI values.

8. (original) The method as described in claim 7 further comprising:

removing one of the previously recorded CPI values and data corresponding to one of the previously identified compatible threads prior to the writing.

9. (original) The method as described in claim 1 further comprising:

writing a first identifier corresponding to the first thread to a compatibility list corresponding to the second thread, wherein the compatibility list stores a plurality of thread identifiers compatible with the second thread.

10. (original) The method as described in claim 9 further comprising:

writing a timestamp corresponding to the first identifier, the timestamp indicating a time at which the time interval occurred, wherein each of the plurality of thread identifiers also include a plurality of timestamps indicating when each of the threads executed with the second thread.

11. (original) The method as described in claim 10 further comprising:

periodically cleaning a plurality of compatibility lists, including the second thread's compatibility list, the cleaning including:

reading the entries corresponding to each of the threads listed in the compatibility lists;

comparing the timestamps listed in the compatibility list with a current time;

determining, based on the comparison, whether the entry associated with the timestamp is stale; and

removing the entry in response to determining that it is a stale entry.

12. (original) The method as described in claim 1 further comprising:

sensing that either the first thread or the second thread is about to complete;

scheduling a new thread to execute, the scheduling comprising:

identifying a compatible thread, the compatible thread being compatible to the tread that is not about to complete;

determining whether the compatible thread is ready to execute; and

dispatching the compatible thread to execute on the SMT processor.

13. (original) The method as described in claim 12 wherein the thread that is about to complete and the compatible thread are listed in a first run queue and wherein the thread that is not about to complete is listed in a second run queue.

14. (currently amended) A computer-implemented method of dispatching software threads to execute on a Simultaneous Multi-Threading (SMT) ~~an SMT~~ processor, said method comprising:

sensing that a completing thread ~~that~~ is about to complete execution on the SMT processor;

identifying a running thread that is still executing on the SMT processor;

checking a list of one or more compatible threads, wherein the compatible threads are compatible with the running thread, and wherein the compatibility is based on a comparison of a first cycles per instruction (CPI) value corresponding to the running thread with one or more CPI values corresponding to each of the listed compatible threads;

determining that one of the compatible threads is ready to execute; and

dispatching the determined thread to execute on the SMT processor.

15. (original) The method as described in claim 14 wherein the completing thread and the compatible threads are listed in a first run queue and wherein the running thread is listed in a second run queue.

16. (currently amended) The method as described in claim 14 wherein the determination that one of the compatible threads is ready to execute further comprises:

checking whether the compatible threads are ready to execute in order of [[a]] the one or more CPI ~~value~~ values corresponding to each of the listed compatible

threads, ~~wherein the compatible threads are checked in an order determined by a CPI value corresponding to each of the compatible threads~~, so that the compatible threads with ~~better~~ corresponding CPI values that are closer to the running thread's CPI value are checked ~~first~~ before the compatible threads with corresponding CPI values that are farther from the running thread's CPI value.

17. (currently amended) An information handling system comprising:

one or more Simultaneous Multi-Threading (SMT) ~~SMT~~ processors;

a memory accessible by the processors;

a compatibility tool for identifying compatible threads to execute on one of the SMT processors, the compatibility tool comprising software code effective to:

identify a time interval during which both a first thread and a second thread are executing on the SMT processor;

retrieve a performance value that is a cycles per instruction (CPI) value and that occurred during the identified time interval;

determine, based upon the retrieved performance value, whether the first thread is compatible to the second thread, wherein the determination includes comparing the CPI value to a threshold value, wherein the first thread and second thread are determined to be compatible if the CPI value is better than the threshold value; and

record the compatibility of the first thread to the second thread in response to the determination.

18. (canceled)

19. (currently amended) The information handling system as described in claim 17 ~~18~~, wherein the CPI value is computed using software code effective to:

retrieve a number of cycles value indicating the number of cycles that occurred during the time interval;

retrieve a number of instructions value indicating the number of instructions that were executed during the time interval; and

divide the number of cycles value by the number of instructions value, the dividing resulting in the CPI value.

20. (canceled)

21. (currently amended) The information handling system as described in claim 17 ~~18~~ further comprising software code effective to:

write a first identifier corresponding to the first thread and the CPI value to a compatibility list that corresponds to the second thread.

22. (original) The information handling system as described in claim 21 wherein the writing is performed in response to software code effective to identify an empty field in the second thread's compatibility list.

23. (original) The information handling system as described in claim 21 wherein the writing is performed in response to software code effective to:

compare the CPI value to one or more previously recorded CPI values that correspond to one or more previously identified compatible threads; and

determine that the CPI value is better than at least one of the previously recorded CPI values.

24. (original) The information handling system as described in claim 23 further comprising software code effective to:

remove one of the previously recorded CPI values and data corresponding to one of the previously identified compatible threads prior to the writing.

25. (original) The information handling system as described in claim 17 further comprising software code effective to:

write a first identifier corresponding to the first thread to a compatibility list corresponding to the second thread, wherein the compatibility list stores a plurality of thread identifiers compatible with the second thread.

26. (original) The information handling system as described in claim 25 further comprising software code effective to:

write a timestamp corresponding to the first identifier, the timestamp indicating a time at which the time interval occurred, wherein each of the plurality of thread identifiers also include a plurality of timestamps indicating when each of the threads executed with the second thread.

27. (original) The information handling system as described in claim 26 further comprising software code effective to:

periodically clean a plurality of compatibility lists, including the second thread's compatibility list, the cleaning comprising software code effective to:

read the entries corresponding to each of the threads listed in the compatibility lists;

compare the timestamps listed in the compatibility list with a current time;

determine, based on the comparison, whether the entry associated with the timestamp is stale; and

remove the entry in response to determining that it is a stale entry.

28. (original) The information handling system as described in claim 17 further comprising software code effective to:

sense that either the first thread or the second thread is about to complete;

schedule a new thread to execute, the scheduling comprising software code effective to:

identify a compatible thread, the compatible thread being compatible to the tread that is not about to complete;

determine whether the compatible thread is ready to execute; and

dispatch the compatible thread to execute on the SMT processor.

29. (original) The information handling system as described in claim 28 wherein the thread that is about to complete and the compatible thread are listed in a first run queue and wherein the thread that is not about to complete is listed in a second run queue.

30. (currently amended) An information handling system comprising:

one or more Simultaneous Multi-Threading (SMT)processors;

a memory accessible by the processors;

a dispatching tool for dispatching compatible threads to execute simultaneously on one of the SMT processors, the dispatching tool comprising software code effective to:

sense that a completing thread ~~that~~ is about to complete execution on the SMT processor;

identify a running thread that is still executing on the SMT processor;

check a list of one or more compatible threads, wherein the compatible threads are compatible with the running thread, and wherein the compatibility is based on a comparison of a first cycles per instruction (CPI) value corresponding to the running thread with one or more CPI values corresponding to each of the listed compatible threads;

determine that one of the compatible threads is ready to execute; and

dispatch the determined thread to execute on the SMT processor.

31. (original) The information handling system as described in claim 30 wherein the completing thread and the compatible threads are listed in a first run queue and wherein the running thread is listed in a second run queue.

32. (currently amended) The information handling system as described in claim 30 wherein the determination that one of the compatible threads is ready to execute further comprises software code effective to:

check whether the compatible threads are ready to execute in order of [[a]] the one or more CPI ~~value~~ values corresponding to each of the listed compatible threads, ~~wherein the compatible threads are checked in an order determined by a CPI value corresponding to each of the compatible threads~~, so that the compatible threads with ~~better~~ corresponding CPI values that are closer to the running thread's CPI value are checked ~~first~~ before the compatible threads with corresponding CPI values that are farther from the running thread's CPI value.

33. (currently amended) A computer program product stored on a computer operable media for identifying compatible software threads to execute on a Simultaneous Multi-Threading (SMT) ~~an SMT~~ processor, said computer program product including instructions that, when executed by an information handling system, causes the information handling system to perform steps comprising:

~~means for~~ identifying a time interval during which both a first thread and a second thread are executing on the SMT processor;

~~means for~~ retrieving a performance value that is a cycles per instruction (CPI) value and that occurred during the identified time interval;

~~means for~~ determining, based upon the retrieved performance value, whether the first thread is compatible with the second thread, wherein the determining includes comparing the CPI value to a threshold value, wherein the first thread and second thread are determined to be compatible if the CPI value is better than the threshold value; and

~~means for~~ recording the compatibility of the first thread with the second thread in response to the determination.

34. (canceled)

35. (currently amended) The computer program product as described in claim 33 ~~34~~ wherein the steps used in computing the CPI value comprise ~~comprises~~:

~~means for~~ retrieving a number of cycles value indicating the number of processing cycles that occurred during the time interval;

~~means for~~ retrieving a number of instructions value indicating the number of instructions that were executed by the SMT processor during the time interval; and

~~means for~~ dividing the number of cycles value by the number of instructions value, the dividing resulting in the CPI value.

36. (canceled)

37. (currently amended) The computer program product as described in claim 33 ~~34~~ wherein the steps further ~~comprising~~ comprise:

~~means for~~ writing a first identifier corresponding to the first thread and the CPI value to a compatibility list that corresponds to the second thread.

38. (original) The computer program product as described in claim 37 wherein the writing is performed in response to identifying an empty field in the second thread's compatibility list.

39. (currently amended) The computer program product as described in claim 37 wherein the ~~means for~~ writing is performed in response to steps comprising:

~~means for~~ comparing the CPI value to one or more previously recorded CPI values that correspond to one or more previously identified compatible threads; and

~~means for~~ determining that the CPI value is better than at least one of the previously recorded CPI values.

40. (currently amended) The computer program product as described in claim 39 wherein the steps further ~~comprising~~ comprise:

~~means for~~ removing one of the previously recorded CPI values and data corresponding to one of the previously identified compatible threads prior to the writing.

41. (currently amended) The computer program product as described in claim 33 wherein the steps further ~~comprising~~ comprise:

~~means for~~ writing a first identifier corresponding to the first thread to a compatibility list corresponding to the second thread, wherein the compatibility list stores a plurality of thread identifiers compatible with the second thread.

42. (currently amended) The computer program product as described in claim 41 wherein the steps further ~~comprising~~ comprise:

~~means for~~ writing a timestamp corresponding to the first identifier, the timestamp indicating a time at which the time interval occurred, wherein each of the plurality of thread identifiers also include a plurality of timestamps indicating when each of the threads executed with the second thread.

43. (currently amended) The computer program product as described in claim 42 wherein the steps further ~~comprising~~ comprise:

~~means for~~ periodically cleaning a plurality of compatibility lists, including the second thread's compatibility list, the cleaning including steps of:

~~means for~~ reading the entries corresponding to each of the threads listed in the compatibility lists;

~~means for~~ comparing the timestamps listed in the compatibility list with a current time;

~~means for~~ determining, based on the comparison, whether the entry associated with the timestamp is stale; and

~~means for~~ removing the entry in response to determining that it is a stale entry.

44. (currently amended) The computer program product as described in claim 33 wherein the steps further ~~comprising~~ comprise:

~~means for~~ sensing that either the first thread or the second thread is about to complete;

~~means for~~ scheduling a new thread to execute, the scheduling comprising:

~~means for~~ identifying a compatible thread, the compatible thread being compatible to the tread that is not about to complete;

~~means for~~ determining whether the compatible thread is ready to execute; and

~~means for~~ dispatching the compatible thread to execute on the SMT processor.

45. (original) The computer program product as described in claim 44 wherein the thread that is about to complete and the compatible thread are listed in a first run queue and wherein the thread that is not about to complete is listed in a second run queue.

46. (currently amended) A computer program product stored on a computer operable media for dispatching software threads to execute on a Simultaneous Multi-Threading (SMT) ~~an SMT~~ processor, said computer program product including instructions that, when executed by an information handling system, causes the information handling system to perform steps comprising:

~~means for~~ sensing that a completing thread ~~that~~ is about to complete execution on the SMT processor;

~~means for~~ identifying a running thread that is still executing on the SMT processor;

~~means for~~ checking a list of one or more compatible threads, wherein the compatible threads are compatible with the running thread, and wherein the compatibility is based on a comparison of a first cycles per instruction (CPI) value

corresponding to the running thread with one or more CPI values corresponding to each of the listed compatible threads;

~~means for~~ determining that one of the compatible threads is ready to execute; and

~~means for~~ dispatching the determined thread to execute on the SMT processor.

47. (original) The computer program product as described in claim 46 wherein the completing thread and the compatible threads are listed in a first run queue and wherein the running thread is listed in a second run queue.

48. (currently amended) The computer program product as described in claim 46 wherein the determination that one of the compatible threads is ready to execute further comprises steps of:

~~means for~~ checking whether the compatible threads are ready to execute in order of [[a]] the one or more CPI ~~value~~ values corresponding to each of the listed compatible threads, ~~wherein the compatible threads are checked in an order determined by a CPI value corresponding to each of the compatible threads~~, so that the compatible threads with ~~better~~ corresponding CPI values that are closer to the running thread's CPI value are checked ~~first~~ before the compatible threads with corresponding CPI values that are farther from the running thread's CPI value.